6800/ICCD

JOURNAL

INTERNATIONAL COMPUTER CENTER DIRECTOR

Volume 1, No. 2 August, 1978

ICCD JOURNAL

The ICCD Journal is published four times yearly by the International Computer Center Director, P.O. Drawer 2790, Norman, Oklahoma 73070. The ICCD is a wholly owned division of Academic World Incorporated. Subscription rates are $18/year U.S. and Canada for membership, $21/year for countries outside the U.S. and Canada.

Opinions expressed by the authors are not necessarily those of the ICCD. Address all subscription correspondence to:

ICCD
P.O. Drawer 2790
Norman, Oklahoma 73070
CABLE: ACAWORLD
405-364-1119

Articles and high quality manuscripts are welcomed by readers of ICCD. ICCD provides a free one-year membership to successful submitters.

TABLE OF CONTENTS

COMING ARTICLES

Emerson Brooks—	Selectric Interface with SWTPC 6800 and Dura Mach 10 or Similar Machines
John A. Waldvogel—	Sperry Univac 0679-02 Terminals (Syner-Beta, Gould et al.)
John A. Waldvogel—	Cursor Block to Cursor Underline Modification to CT 10-64 TVT II
Ron Hilbun—	Crystal Modification to SWTPC MPA/2 Processor versus R/C Circuit

COMING EVALUATIONS

SWTPC 3.0 Basic
TSC Flex+ + + 1.0
TSC Word Processor System 2.3
SWTPC EPROM System
SWTPC MF-D Large Disc System
Smoke Signal BFD-68 Floppy Interface and DOS

COMMENTS FROM THE PUBLISHER

The 6800 ICCD is now running. Soon the CHIP will be mailed and with some degree of regularity you subscription will be maintained. Lots of exciting things are happening in the world of the 6800. Before we mention the excitement, perhaps we should speak of some caution and needs to protect the 6800 world. Outside of the 6800 universe there is a galaxy producing Pet and Radio Shack TRS-80 Computers. The 6800 computer manufacturers by-in-large are few in number and do not approach the super bigness of Pet and Radio Shack. The ICCD was formed to serve the 6800 user. We believe that ICCD can be the vehicle to promote and disseminate the advantages which outway other systems. Since our first communication, SWTPC and Smoke Signal have joined to assist you through their support of ICCD. Others will follow. If you have personal interface with any other manufacturers, please send them this way. We will be asking shortly for your assistance in preparing a master registry of all 6800 users. This will solidify our communications as well as pass on innovations without the usual delays encountered. A ICCD test console is being constructed—when complete pictures will appear in the Journal or CHIP.

Your Editor recently visited and saw the assembly line of the Microcomputer Devices Inc., "*Selectra-Term.*" Impressive and built from brand-new (shipped from IBM) 15'' Selectric II Typewriters. If you are interested give Shelly Howard a call.

Be sure to tell us what you need and we will get those articles which meets the need of the membership. We have a large disc system on the way from SWTPC and will tell you more after we go over the system and put it through its paces. If there is one thought your editor would like to leave—that is we in the 6800 universe must join together.

We welcome a small contingency of new members from the Computer Information Exchange (formerly the SS-50 Buss).

Harold Zallen, Ph.D.
Editor-Publisher

MEMTEST ALTERED FOR SWTPC SWTBUG ®

Emerson Brook's MEMTEST appearing in the first issue of the Journal 6800 ICCD has been quite well received. Our readers have asked that we provide them with an alteration for use with SWTPC SWTBUG®. In response to these requests the program is listed below. This program has been tested and appears to function properly. To assemble, an EQU for MCL must be included in the EQUATE section.

Acknowledgement for the original work goes to Dr. Emerson Brooks.

Jack D. Johnson
2816 Wood Creek Road
Midwest City, OKLAHOMA 73110


```
                      NAM    MEMTST
                    * MEMTEST ALTERED FOR SWTPC SWTBUG

                    * BY DR. EMMERSON BROOKS

                    * REVISED BY JACK D. JOHNSON


            A002    MEMBEG EQU   $A002    TEST START ADDRESS
            A004    MEMEND EQU   $A004    TEST END ADDRESS
            A000    BSTORE EQU   $A000    INITIAL LOAD VALUE
            A00E    WRONG  EQU   $A00E    INCORRECT VALUE
            A00F    RIGHT  EQU   $A00F    CORRECT VALUE
            A010    BADMEM EQU   $A010    ERROR ADDRESS
            000A    LF     EQU   $0A      LINE FEED
            000D    CR     EQU   $0D      CARRIAGE RETURN
            0008    BS     EQU   $08      BACK SPACE
            E0CC    OUTS   EQU   $E0CC    OUTPUT SPACE
            E1D1    OUTEEE EQU   $E1D1    PRINT CHARACTER
            E0CA    OUT2HS EQU   $E0CA    PRINT 2 HEX, SPACE
            E0C8    OUT4HS EQU   $E0C8    PRINT 4 HEX, SPACE
            E07E    PDATA1 EQU   $E07E
            E19D    MCL    EQU   $E19D    END OF LINE
                           OPT   O,NOS,NOG,P
      A014                 ORG   $A014
      A014 FF A010 ERROR   STX   BADMEM   PRINT ERROR MESSAGE
      A017 B7 A00E         STA A WRONG
      A01A F7 A00F         STA B RIGHT
      A01D CE E19D         LDX   #MCL     CR, LF
      A020 BD E07E         JSR   PDATA1
      A023 CE A00E         LDX   #WRONG   POINT TO DATA
      A026 BD E0CA         JSR   OUT2HS   PRINT INCORRECT VALUE
      A029 BD E0CA         JSR   OUT2HS   PRINT CORRECT VALUE
      A02C BD E0C8         JSR   OUT4HS   PRINT ERROR ADDRESS
      A02F FE A010         LDX   BADMEM   RESTORE INDEX
      A032 20 35           BRA   RETURN
      A04A                 ORG   $A04A
      A04A F6 A000 START   LDA B BSTORE   LOAD MEMORY
      A04D FE A002         LDX   MEMBEG
      A050 E7 00   LOOP1   STA B 0,X
      A052 BC A004         CPX   MEMEND
      A055 27 04           BEQ   CHECK    END ADDRESS?
      A057 08              INX
      A058 5C              INC B
      A059 20 F5           BRA   LOOP1
      A05B BD E0CC CHECK   JSR   OUTS     MOVE CURSOR
      A05E F6 A000         LDA B BSTORE   TEST MEMORY
      A061 FE A002         LDX   MEMBEG
      A064 A6 00   LOOP2   LDA A 0,X
      A066 11              CBA
      A067 26 AB           BNE   ERROR    GO PRINT MESSAGE
      A069 BC A004 RETURN  CPX   MEMEND
      A06C 27 04           BEQ   CYCLE    END ADDRESS?
      A06E 08              INX
      A06F 5C              INC B
      A070 20 F2           BRA   LOOP2
      A072 86 08   CYCLE   LDA A #BS      MOVE CURSOR BACK
      A074 BD E1D1         JSR   OUTEEE
      A077 7C A000         INC   BSTORE   INCR. INITIAL LOAD
      A07A 20 CE           BRA   START    DO ANOTHER TEST
                           END

TOTAL ERRORS 00000
```

TINY WORD PROCESSOR

Here is a word processor Version N. (N). It has a lot of nice features. Try it out. It is getting larger each time the author uses it. It numbers lines and gives instructions, etc.

```
0005 REM ++WORD PROCESSOR FOR SWTPC++
0010 REM +++JIM CALDWELL+++
0020 GOSUB 080
0025 PRINT "SET STRING => LINE LENGTH"
0028 REM +++OUTPUT PORT+++
0030 INPUT "PRINTER PORT",P
0035 INPUT "PRINTER LINE WIDTH",W
0040 INPUT "LENGTH AND NUMBER OF LINES",
L,N
0055 REM +++HOME & CLEAR+++
0060 GOSUB 080
0070 GOTO 150
0080 PRINT CHR$(16);CHR$(22)
0090 RETURN
0100 PRINT TAB(L+3);"?"
0105 REM +++VERTICAL TAB UP+++
0110 PRINT CHR$(11);
0120 IF L < 31 THEN 140
0130 PRINT CHR$(11);
0140 RETURN
0150 DIM I(N),A$(N),J$(N)
0160 INPUT "NEED INSTRUCTIONS",G$
0170 IF ASC(G$)=78THEN220
0180 PRINT "TYPE TEXT BETWEEN  THE QUEST
ION"
0190 PRINT "MARKS.   DO NOT USE COMMAS !
!!!"
0200 PRINT "IF LAST LINE IS SHORT ADD SP
ACE"
0210 PRINT "TYPE DONE  IF LINES NOT = TO
 ";N
0215 PRINT "TYPE STOP WHEN FINISHED AND"
0216 PRINT "SYSTEM ASKS   'HARD COPY'"
0220 GOSUB 100
0230 FOR I=1TON
0240 INPUT A$(I)
0250 IF A$(I)="DONE"THEN690
0260 GOSUB 100
0270 NEXT I
0280 GOTO 360
0290 REM +++EDIT+++
0300 GOSUB 080
0310 FOR I=1TON
0320 PRINT (I);A$(I)
0330 NEXT I
0340 GOTO 620
0350 IF ASC(G$)=89THEN620
0360 REM +++JUSTIFY+++
0370 GOSUB 080
0380 FOR I=1TON
0385 GOSUB 800
0390 NEXT I
```

James D. Caldwell, K5OHU
1001 Trout Avenue
Port Isabel, Texas 78578

Editor's Note: Jim has other revisions...we will publish these later.

THE TSC WORD PROCESSOR

These few short words should not be construed as a formal review. *That will come later.* Our initial reaction to the TSC Editor and Text Processor is highly favorable. We have tested in some ways the system but as yet have not given the system a chance to show all it can do. We caution you to read the documentation thoroughly. It is truly a sophisticated system and the price should not fool you. Yes, other similar systems are in the *thousands of dollars range.* We plan to put all of our typed copy onto the TCS System and as we develop our techniques we will share them with our readers. For those who have not arrived in the disc based systems, keep in mind that *FLEX+++* another innovation for the 6800 family is a Disc Operation System with many, many wrinkles. Dave Shirk, President of TSC has asked the users of *FLEX+++* whether they should support a user's group. If TSC does not choose to, ICCD will be glad to assist. We like *FLEX +++* and the entire Word Processing System by TSC. Keep up that good software for 6800's. Smoke Signal Owners, R.L. Smith (in this issue) has a means for using TSC with your System. SWTPC you will soon have a similar patch for the Minifloppy Disc System.

Editor-

FDOS SPACE SAVER

You input the name; starting/ending address and it will print out, on a PR-40, a list of programs with the actual amount of program space that each program requires.

This program determines the number of sectors on a minifloppy diskette to save for a program without allowing for a 25% buffer like the floppy disk operating system does.

The two hex addresses are converted to base 10. The difference is computed and divided by 256 to determine how many sectors are required. This value is changed into a hex a decimal number as required by the floppy disk operating system (FDOS). The reason for all of this number swapping, is the fact that the FDOS assigns a 25% additional buffer to each program space on the diskette to allow for program expansion. Which is great while developing programs. But also means that up to 25% of the 90K (22K) of the disk is full of nulls.

The author uses one diskette for a scratch pad and when a program is finished (***someday a program has to be finished***) a space can be 'created' that does not include the 25% buffer. In the example given the FDOS assigned 12 hex sectors to a OD90 hex program (thats 18 base 10 sectors for a 3483 base 10 program) this program calculates that only one hex sectors (14 base 10) were needed.

This results in an additional 4 (base 10 sectors or one K of storage space that is available on the diskette.

The program should run in any 8K basic that has string handling capabilities and parts of it should be useful to anyone that uses hex notation and doesn't have 16 fingers to count on.

Line 500 is the home and clear for the SWTPC CT-1024®. This can be changed for other terminals.

Line 1100 breaks the 4 digit hex a decimal address down into 4 individual values. By using the MID$() function to change them into A$; B$; C$; and D$.

Sub routine 2000 checks for alpha characters and assigns a decimal value to N *i.e.*, A-10, B-11, C-12, D-13, E-14 and F-15. If they are not alpha values then the actual value of N$ is assigned to N with the VAL() function.

Continued on page 4

Continued from page 3

Each digit is converted from a base 16 notation to a base 10 notation by multiplying each digit by the decimal value of each hex digit.

The MSB of a 4 digit hex value if equal to that value times 4096. The next is equal to times 256. The next to times 16 and the lsb times 1.

The program adds these decimal values stores them in F(1) after clearing the A$ to N$ variables by assigning a value of " (*which is nothing*) and a value of 0 to N.

The next hex a decimal address is converted and stored in F(2).

At line 3000 the difference is computed and divided by 256 to determine the number of sectors *base 10* to allocate for the program.

Sub routine 7000 rounds up the number of sectors to an even value (*partial sectors cannot be assigned*).

Starting at line 3070 the decimal number of sectors required are converted to hex decimal which is what the FDOS requires.

The hex decimal equivalent is found by dividing by 16 and using the INT() to place a value on the first hex digit and converting this value to a string value with F$=STR$(F) at line 3075.

The second hex decimal digit is the remainder converted to hex idecimal notation if greater than 9 *i.e.*, A-B-C-D-E-F.

These values are stored and the system set up to calculate another value or end the program.

Files

Boats 16 03 12 55 2720 34BO 6EOO

```
FDOS READY

  $DOS      00 00 14 00 2400 2FFF 2600

  CMTEST    03 00 03 55 2720 2858 6E00

  WORDS     03 03 08 55 2720 2F70 6E00

  BJACK     04 04 08 22 0070 0646 0100

  SEC/LIST  05 02 0C 55 2720 2F44 6E00

  NAVTEST   06 04 06 55 2720 2858 6E00

  SPD/CAL   07 00 03 55 2720 29C7 6E00

  BREAD     07 03 06 55 2720 2AC7 6E00

  TWRITE    07 09 02 55 2720 278E 6E00

  SEARCH    08 01 04 55 2720 2919 6E00

  EDITOR    08 05 0E 22 0100 085F 0100

  ASSEM     09 09 1F 22 0080 17FF 0100

 SECTORS REMAINING (HEX)  00E6

HEX/SEC LISTING

BREAD     2720    2A27    04   SECTORS
TWRITE    2720    2783    01   SECTORS
SEARCH    2720    2919    02   SECTORS
EDITOR    0100    085F    0B   SECTORS
ASSEM     0100    17FF    17   SECTORS
BJACK     0070    0646    06   SECTORS
```

```
0005 REM *** JIM CALDWELL K5OHU ***
0010 REM *** HEX CONVERSION FOR SWTPC **
*
0020 REM *** MINIFLOPPY 256 BYTE SECT **
*
0030 GOSUB 500
0040 PRINT "HEX/SECTOR CALCULATOR"
0200 FOR L= 1 TO 40
0210 INPUT "NAME OF PROGRAM",L$(L)
0220 LET X=0
0230 GOTO 1000
0500 REM *** HOME AND CLEAR CT-1024 ***
0510 PRINT CHR$(16);CHR$(22);CHR$(00);
0520 RETURN
1000 REM ***      INPUT     ***
1005 LET H$=H$(L)
1010 INPUT "FIRST HEX ADDRESS",H$(L)
1015 LET H$=H$(L)
1020 LET T = 0
1030 DIGITS= 0
1100 REM *** START CONVERSION ***
1110 LET A$ = MID$(H$,1,1)
1120 LET B$ = MID$(H$,2,1)
1130 LET C$ = MID$(H$,3,1)
1140 LET D$ = MID$(H$,4,1)
1200 REM *** CHANGE HEX TO NUMERIC ***
1210 LET N$=A$
1220 GOSUB 2000
1230 LET T = N*4096
1240 LET N$ = B$
1250 GOSUB 2000
1260 LET T = T+(N*256)
1270 LET N$ = C$
1280 GOSUB 2000
1290 LET T = T+(N*16)
1300 LET N$ = D$
1310 GOSUB 2000
1320 LET T = T+N
1400 REM ***      PRINT     ***
1420 LET A$=""
1430 LET B$=""
1440 LET C$=""
1450 LET D$=""
1460 LET H$=""
1470 LET N$=""
1480 LET N = 0
1500 REM *** SET UP NEXT NUMBER ***
1510 LET X = X+1
1520 LET F(X) = T
1530 IF D = 1 THEN 3000
1540 LET D = 1
1550 REM ***  GET NEXT HEX  ***
1560 INPUT "NEXT  HEX ADDRESS",I$(L)
1565 LET H$=I$(L)
1570 GOTO 1020
2000 REM *** CONVERSION ***
2010 IF N$ = "A" THEN N = 10
2020 IF N$ = "B" THEN N = 11
2030 IF N$ = "C" THEN N = 12
2040 IF N$ = "D" THEN N = 13
2050 IF N$ = "E" THEN N = 14
2060 IF N$ = "F" THEN N = 15
2070 IF N<10 THEN 2200
2080 RETURN
2200 LET N = VAL(N$)
2210 RETURN
3000 REM *** COMPUTE DIFFERENCE ***
3010 PRINT
3025 PRINT
3030 LET D=F(2)-F(1)
3045 GOSUB 7000
3070 LET F = INT(D(1)/16)
3075 LET F$(L)=STR$(F)
3080 LET S=D(1)-(16*INT(D(1)/16))
3090 IF S > 9 THEN 5000
3095 LET S$(L)=STR$(S)
4000 REM * HEX SEC *
4010 GOTO 6000
4100 GOTO 6100
5000 REM *** BACK TO HEX ***
5010 IF S = 10 THEN S$="A"
5020 IF S = 11 THEN S$="B"
5030 IF S = 12 THEN S$="C"
5040 IF S = 13 THEN S$="D"
5050 IF S = 14 THEN S$="E"
5060 IF S = 15 THEN S$="F"
5070 LET S$(L)=S$
6000 REM * FINISH *
6100 INPUT "ANOTHER CALCULATION",C$
6105 GOSUB 500
6110 IF LEFT$(C$,1)="N" THEN 8000
6130 NEXT L
6500 RUN
7000 REM *** ROUND UP SECTORS ***
7010 IF D<256 THEN 7100
7020 IF D/256 > INT(D/256) THEN 7200
7030 LET D(1)=INT(D/256)
7040 RETURN
7100  D(1) = 1
7110 RETURN
7200  D(1)=INT(D/256)+1
7210 RETURN
8000 REM ** LIST OF SECTORS **
8005 INPUT "OUTPUT PORT",0
8010 FOR P = 1 TO L
8015 PORT= 0
8020 PRINT L$(P);TAB(10);H$(P);"    ";I$(
P);"    ";F$(P);S$(P);"  SECTORS"
8030 NEXT P
8040 PORT= 1
8050 INPUT "ANOTHER LIST",C$
8060 IF LEFT$(C$,1)="Y" THEN 6500
8070 CATALOG
```

This is a sample of the files function for the SWTPC® minifloppy disk system first the program name; then the track where the program starts; the starting sectors; the number of sectors allocated for the program (including the 25% buffer). The 55 means a basic program then the starting and ending address the 6EOO is a system code.

The sample printout shows how much diskette space is saved by only creating the actual space needed.

James D. Caldwell, K5OHU
1001 Trout Avenue
Port Isabel, Texas 78578

HANDY DANDY LOAN-BALANCE PROGRAM

for determining loan balances.

Here is a little program, although it is not a large program do not underestimate its power. *Be sure to input interest as an annual rate.* Note Line 3000. This has been included to allow you to type from your keyboard to simulate typewriter action of your PR-40. This print program may be ad-

Continued on page 5

Continued from page 4

ded to other programs to allow the addition of text or comments at the end of the program or to add an instruction list. The ease of its use allows these additions without loading up the word processor yet you still have the capability of editing. CAUTION: You must not use a carriage return.

```
LOAN BALANCE     500.00
INTEREST RATE    9.25 %
MONTHLY PAYMENT  50.00
FIRST MONTH      SEPTEMBER
MONTH    PAYMT INTST PRINC BALANCE
1 SEP    50.00 3.85  46.14 453.85
2 OCT    50.00 3.49  46.50 407.35
3 NOV    50.00 3.14  46.85 360.49
4 DEC    50.00 2.77  47.22 313.27
5 JAN    50.00 2.41  47.58 265.68
6 FEB    50.00 2.04  47.95 217.73
7 MAR    50.00 1.67  48.32 169.41
8 APR    50.00 1.30  48.69 120.71
9 MAY    50.00 0.93  49.06 71.64
10 JUN   50.00 0.55  49.44 22.20
11 JUL   22.37 0.17  22.20 0.00
```

```
0001 GOSUB 50
0005 INPUT "PRINTER PORT",P
0020 GOTO 100
0050 PRINT CHR$(16);CHR$(22);
0060 RETURN
0100 INPUT "LOAN BALANCE    ",B
0110 INPUT "INTEREST RATE   ",R
0120 INPUT "MONTHLY PAYMENT ",A
0130 INPUT "FIRST MONTH     ",F$
0131 PORT= P
0132 PRINT "LOAN BALANCE    ";B
0134 PRINT "INTEREST RATE   ";R;"%"
0136 PRINT "MONTHLY PAYMENT ";A
0138 PRINT "FIRST MONTH     ";F$
0140  R=R/12/100
0150 RESTORE
0160 READ M$
0170 IF LEFT$(F$,3)=M$THEN190
0180 GOTO 160
0190 PRINT "MONTH   PAYMT INTST PRINC BA
LANCE"
0200  M=1
0210  I=R*B
0220  P=A-I
0230  B=B-P
0240 GOSUB 296
0250  M=M+1
0255 READ M$
0257 IF M$="END"THEN259
0258 GOTO 261
0259 RESTORE
0260 GOTO 255
0261 IF B < P THEN 290
0270 GOTO 210
0290  A=B+(B*R)
0292  I=B*R
0294  P=B
0295  B=0
0296 DIGITS= 0
0300 PRINT M;
0305 DIGITS= 2
0306 PRINT M$;TAB(9);A;TAB(15);I;TAB(21)
;P;TAB(27);B
0309 IF B=0THEN500
0310 RETURN
0400 DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AU
G,SEP,OCT,NOV,DEC,END
0500 PORT= 1
0510 END
3000 INPUT A$:PRINT#7,TAB(5);A$:GOTO3000
```

James D. Caldwell, K5OHU
1001 Trout Avenue
Port Isabel, Texas 87858

BAUD SELECTION IMPROVEMENT FOR SWTPC-6800 ®

After using the SWTPC 6800®/CT-1024/AC-30® combination for several months, it was decided that improvements were needed which would allow the following:

The single baud rate selector switch, which is conveniently located on the CT-1024 or CT-64 terminal, should select a common rate for both the terminal and the computer (no need to switch two switches). Automatic selection to the 300 baud rate should be provided when writing to or reading from the AC-30® cassette interface (no need to remember to switch the selector to 300 when recording).

A simple modification, performed only to the AC-30® cassette interface, can accomplish both the above features without loss of any of its existing capabilities.

How it works

The computer clock output is strapped (within the computer) for 300 baud. The terminal clock output is selectable from 150 to 1200 baud via its baud selector switch. Both of these clocks are brought into the AC-30® and are available for use, their traces where they enter the board are cut which frees them from their former functions.

A jumper is not attached between pins 3 and 5 of IC6B and this point is utilized as a common clock input for the terminal and computer.

The two clocks which were isolated when their traces were cut are fed as inputs to a new 7400 chip which is wired as a two input data selector whose output feeds IC6B, pins 3/5. The control input to this data selector is connected, via 2 diodes to the collectors of Q5 and Q6, in such a manner that if either the record ready or the read ready lamp is lit, 300 baud is furnished to the terminal and computer, however, if neither is lit, the variable baud rate is selected.

Modification Procedures

1. Install 7400 chip, 1K resistor, 2 ea., diodes and .05 or .1 Mfd capacitor on Radio Shack #276-024 socket adapter per drawing (note that the socket adapter will accomodate 16 pins and the 7400 only had 14. The diodes terminate on these extra pads.)
2. Remove the top cover from the AC-30®, be sure the AC-30® is unplugged because the location of its fuse represents a chip hazard as well as a safety hazard when making this modification.
3. Viewing the AC-30® PC board from the top rear position, locate the trace going from the anode of D-8 to the computer connector (second pin from the right) and cut this trace between the connector and D-8.
4. While still viewing the AC-30 board from the top rear, locate the trace going from the center connector, (Terminal 16X clock output) 5th pin from left end, which goes toward the front of the board. Cut this trace between the points where passed under C-21 and R-45.
5. Connect a jumper from the anode of D-10 to the trace going to the anode of D-3 (this trace disappears under the end of IC6, at an angle).
6. Mount the socket adapter assembly, complete with its installed components, inside the AC-30®. (I mounted mine by soldering a 1 1/2" stiff bare hookup wire to the wider copper grounded area surrounding IC16 and another to the 5 volt feedthru point located between the ends of IC9 and IC8, then soldering the grounded wire to the #7 pad and the 5 volt wire to the #14 pad of the chip socket adapter, with the bottom side of the adapter facing upward).
7. Connect a jumper wire from the cathode of one of the diodes mounted on the chip board to the collector of Q-5 (which is connected to one end of R-30). Connect another jumper wire from the other diode cathode to the collector of Q-6 (which is connected to one end of R-31).
8. Connect a jumper wire from pad #8 of the chip adapter to the

Continued on page 6

Continued from page 5

anode of D-10 (you already have one jumper connected here).

9. Connect a jumper wire from Pad #12 of the chip adapter to the connector side of the trace cut in step #3.

10. Connect a jumper wire from Pad #5 of the chip adapter to the connector side of the trace cut in step #4.

11. Connect ground wire to pin 7 of chip adapter and +5 V. wire to Pin 14 of chip adapter for power (if not previously connected when mounting the adapter inside the AC-30).

This completes the modification.

Testing After Modification

1. Turn on AC-30®, terminal, and 6800 computer.
2. Set Baud Rate Selector on terminal to 1200 baud.
3. Reset computer, an "*" should come up.
4. Depress "P," memory dump should appear on the screen at a 1200 baud rate.
5. While observing display, switch the *record ready* lamp on. The dump should now be observed on the screen at 300 baud. Turn *record ready* lamp off. The dump should now go to 1200 baud.
6. While observing display, switch the *read ready* lamp on. Again the 300 baud dump should be observed. Turn the *read ready* lamp off.

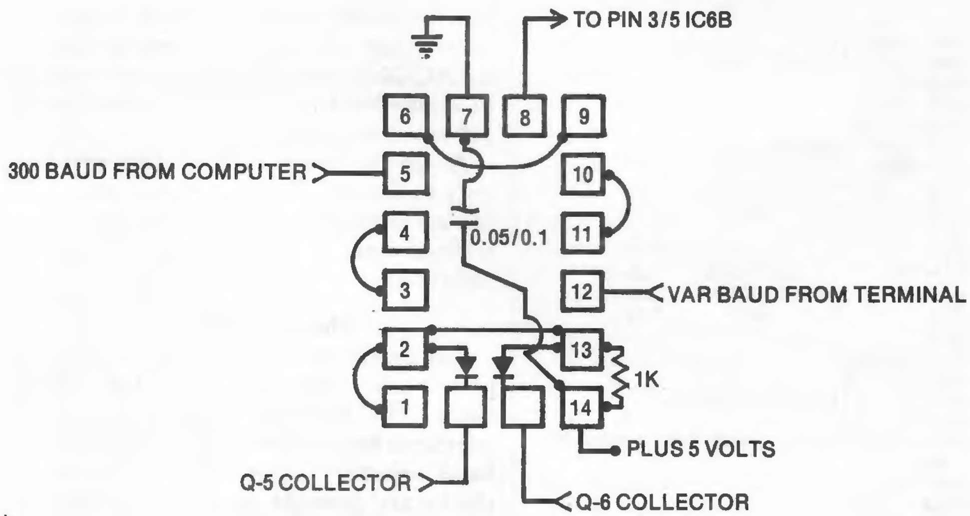
You are now ready to use your modified system.

Bob Dunsmore
6520 N.W. 12th Street
Oklahoma City, Oklahoma 73127

BAUD SELECTION IMPROVEMENT FOR CT 1024/CT-64, SWTP 6800, AC-30 SYSTEM

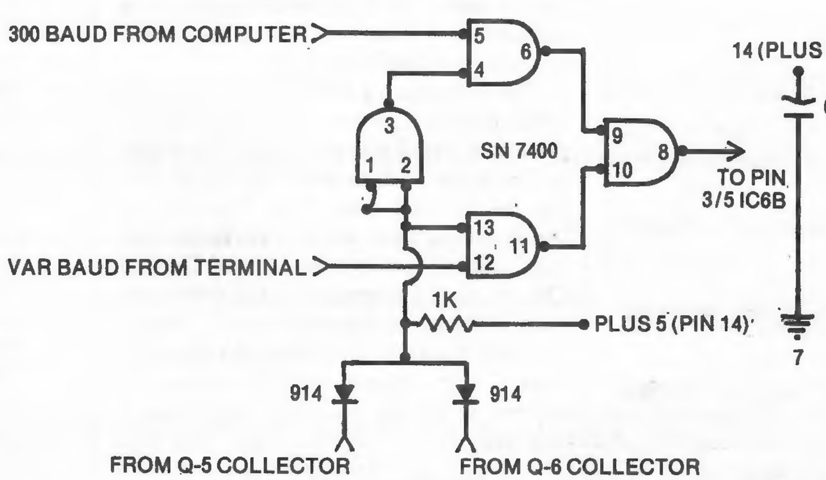
PARTS LIST

1 each SN-7Quad Nand Gate
1 each 1 K Ohm Resistor
2 each 9 (or equiv.) Silicon Diode
1 each 0.05 or 0.1 25 VDC Capacitor
1 each Socket Adapter (Radio Shack or equiv. No. 276-024) Pin



BOTTOM VIEW OF SOCKET ADAPTER WITH 7400 IC CHIP INSTALLED

(NOTE THAT CHIP IS INSTALLED IN ADAPTER SO THAT THE BOTTOM 2 PADS ON THE ADAPTER ARE EMPTY)



6800 DISASSEMBLER

One of the most helpful software tools in any system library is a disassembler. Simply stated, a disassembler translates machine code into assembly language. This translation, of course, cannot insert meaningful statement labels or comments but is often the only way to obtain ANY documentation of a program. Debugging is easier with a source code listing to trace program flow. Programs traded with which are poorly documented prime candidates to be disassembled. Programs which require modification to run on a particular system configuration should first be disassembled. A disassembler is an aid to learn both the style and the philosophy of the author of a particular program.

Since need has been established for a disassembler the author proposed the following which has been adapted from a disassembler by Hughes (1) in 1977. (Several alterations have been made to take advantage of the author's terminal and to change the formatting of the printout from the originally published version). It is suggested that two copies of the disassembler be made. One ORGed to $0100 and the other to $3100 (or elsewhere not on top of your disk or cassette operating system). This will allow disassembly of programs anywhere in core—although some programs may have to be done in two parts.

The program flow is rather straight forward. It is assumed you use MIKBUG® or SWTBUG® and have a parallel printer on port 7. If some other printing capability is to be used, change the printer initialization routine (lines 0340 thru 0380) and the subroutine PRINT (lines 2910 thru 3030). The routine INPUT reads one character from your terminal and places it in the A register. The routine OUTPUT prints a line of text (terminated by a hex 04) on the terminal. The routine BADDR reads a four-digit hex address from the terminal and places it in the index register. MONIT is the address of the system monitor.

The first section of the program inputs the starting and ending addresses for the disassembly and prints a title. The 1 (line 0450) and 2 (line 0520) turn enhanced mode on and off on the author's printer. (Model IP-125, Integral Data System).

The next section attacks the instruction to be disassembled. First the address of the instruction is moved to the print line. This is followed by the machine code instruction. If any part of the machine code is a valid ASCII character, that character is inserted in the printline. Hughes (1) used a table (referenced in line 0790) to hold encripted information about the instruction being disassembled. Each entry contains a one-byte index into a table of mnemonic operation codes, flags set if the A or B registers are referenced, a flag set if the instruction causes an unconditional branch, and the length of the instruction. This format permits relatively easy conversion of this program to handle other instruction sets. This table entry is processed and the mnemonic operation code moved to the print line. The program next determines the addressing format used by the instruction and sends the operands to the print line. Both the relative and absolute addresses are given for relative addressed instructions. The line is then printed. If the use of a statement label is obvious for the next line, a period is inserted for that line. If more instructions are to be processed, the disassembly is repeated. Otherwise the word END is printed and control is given to MIKBUG®/SWTBUG.®

The subroutine CNVT converts the hex digits in a byte to ASCII and stores them in the location referenced by the index register. The subroutine PRINT sends output to a parallel port referenced by PRINTR. The subroutine CHAR is used to insure that only valid ASCII characters are moved to the printline. The padding following the OPCODE table is required because the author's disk system tends to drop data from the beginning and end of files. This may not be a problem for others.

A disassembler cannot tell the difference between data and instructions. This results in data being disassembled. When an invalid operation code is discovered, the disassembler places three asterisks in the operation code field of the printout. When valid operation codes are found in data areas, strange sequences of instructions can be generated. Frequent asterisks are a good clue to large data areas and indicate "instructions" should be ignored. The sample output included here includes part of a data area to demonstrate this weakness. Despite any weaknesses, the source code produced by a disassembler is better than no source code at all!

Now we know what a disassembler is and how this one works. The next step is to use its output to solve a problem. Perhaps the best approach is to use an example. Suppose a program you obtained causes your printer to do strange things and you suspect the problem is caused by an illegal output control character. The first step is to disassemble the program. Since your printer is addressed as Port 7 or $801C, scan the disassembly for "$801C". (If the program is very large, the SWTBUG® FIND instruction may be used to locate all occurances of hex 1C.) The print routine can then be examined to find its logical entry point. A check for BSR and JSR instructions to this point should quickly pinpoint the trouble.

This program should serve as a useful addition to your collection of programming tools.

```
PAGE  001    DISASM


00010                         NAM    DISASM
00020           801C   PRINTR EQU    $801C
00030           E1AC   INPUT  EQU    $E1AC
00040           E07E   OUTPUT EQU    $E07E
00050           E047   BADDR  EQU    $E047
00060           E0E3   MONIT  EQU    $E0E3
00070                         OPT    O,NOS,NOG,P
00080      0100               ORG    $0100
00090      0100 20 61         BRA    START
00100      0102 0002   ADDR   RMB    2
00110      0104 0002   DONE   RMB    2
00120      0106 0002   TEMP   RMB    2
00130      0108 0001   SAVEOP RMB    1
00140      0109 0001   OPRNDA RMB    1
00150      010A 0001   OPRNDB RMB    1
00160      010B 0001   OPINDX RMB    1
00170      010C 0001   OPINFO RMB    1
00180      010D 53     QUESTA FCC    'STARTING ADDRESS'

00190      011B 04            FCB    4
00200      011E 45     QUESTB FCC    'ENDING ADDRESS'
```

Continued on page 9

Continued from page 8

```
00210 012C 04                FCB    4
00220 012D 0D         QUESTC FCB    $D,$A
00230 012F 54                FCC    'TITLE:

00240 0136 04                FCB    4
00250 0137 20         LINE   FCC    44,

00260 0163 CE 010D    START  LDX    #QUESTA
00270 0166 BD E07E           JSR    OUTPUT
00280 0169 BD E047           JSR    BADDR
00290 016C FF 0102           STX    ADDR
00300 016F CE 011E           LDX    #QUESTB
00310 0172 BD E07E           JSR    OUTPUT
00320 0175 BD E047           JSR    BADDR
00330 0178 FF 0104           STX    DONE
00340 017B 7F 801D           CLR    PRINTR+1
00350 017E 86 7F             LDA A  #$7F
00360 0180 B7 801C           STA A  PRINTR
00370 0183 86 3F             LDA A  #$3F
00380 0185 B7 801D           STA A  PRINTR+1
00390 0188 CE 012D           LDX    #QUESTC
00400 018B BD E07E           JSR    OUTPUT
00410 018E 08                INX
00420 018F 86 0C             LDA A  #$C
00430 0191 A7 00             STA A  X
00440 0193 08                INX
00450 0194 86 01             LDA A  #1
00460 0196 A7 00             STA A  X
00470 0198 BD E1AC    TITLE  JSR    INPUT
00480 019B 08                INX
00490 019C A7 00             STA A  X
00500 019E 81 0D             CMP A  #$D
00510 01A0 26 F6             BNE    TITLE
00520 01A2 86 0A             LDA A  #$A
00530 01A4 A7 01             STA A  1,X
00540 01A6 86 02             LDA A  #2
00550 01A8 A7 02             STA A  2,X
00560 01AA 86 04             LDA A  #4
00570 01AC A7 03             STA A  3,X
00580 01AE CE 0137           LDX    #LINE
00590 01B1 BD 03C5           JSR    PRINT
00600 01B4 CE 0137           LDX    #LINE
00610 01B7 86 2B             LDA A  #43
00620 01B9 C6 20             LDA B  #'
00630 01BB E7 00      CLEAR  STA B  X
00640 01BD 08                INX
00650 01BE 4A                DEC A
00660 01BF 26 FA             BNE    CLEAR
00670 01C1 CE 0137    REPEAT LDX    #LINE
00680 01C4 B6 0102           LDA A  ADDR
00690 01C7 BD 0395           JSR    CNVT
00700 01CA B6 0103           LDA A  ADDR+1
00710 01CD BD 0395           JSR    CNVT
00720 01D0 FE 0102           LDX    ADDR
00730 01D3 A6 00             LDA A  X
00740 01D5 B7 0108           STA A  SAVEOP
00750 01D8 A6 01             LDA A  1,X
00760 01DA B7 0109           STA A  OPRNDA
00770 01DD A6 02             LDA A  2,X
00780 01DF B7 010A           STA A  OPRNDB
00790 01E2 CE 03D7           LDX    #OPS
00800 01E5 FF 0106           STX    TEMP
00810 01E8 B6 0108           LDA A  SAVEOP
00820 01EB 48                ASL A              2xOP
00830 01EC 24 03             BCC    AA
00840 01EE 7C 0106           INC    TEMP
00850 01F1 F6 0107    AA     LDA B  TEMP+1
00860 01F4 1B                ABA
00870 01F5 B7 0107           STA A  TEMP+1
00880 01F8 24 03             BCC    BB
00890 01FA 7C 0106           INC    TEMP       #OPS
00900 01FD FE 0106    BB     LDX    TEMP       INDEX
00910 0200 A6 00             LDA A  X
00920 0202 B7 010B           STA A  OPINDX
00930 0205 A6 01             LDA A  1,X
00940 0207 B7 010C           STA A  OPINFO
00950 020A 84 03             AND A  #3
00960 020C 16                TAB
00970 020D FE 0102           LDX    ADDR
00980 0210 08         CC     INX
00990 0211 4A                DEC A
01000 0212 26 FC             BNE    CC
01010 0214 FF 0102           STX    ADDR
01020 0217 CE 2020           LDX    #$2020
01030 021A FF 013F           STX    LINE+8
01040 021D FF 0141           STX    LINE+10
01050 0220 FF 0145           STX    LINE+14
01060 0223 CE 013D           LDX    #LINE+6
01070 0226 B6 0108           LDA A  SAVEOP
01080 0229 BD 0395           JSR    CNVT
01090 022C B6 0108           LDA A  SAVEOP
01100 022F BD 03CC           JSR    CHAR
01110 0232 A7 05             STA A  5,X         store OP in line
01120 0234 5A                DEC B              decr. no. bytes
01130 0235 27 1F             BEQ    DD
01140 0237 B6 0109           LDA A  OPRNDA
01150 023A BD 0395           JSR    CNVT
01160 023D B6 0109           LDA A  OPRNDA
01170 0240 BD 03CC           JSR    CHAR
01180 0243 A7 04             STA A  4,X
01190 0245 5A                DEC B              decr. no. bytes
01200 0246 27 0E             BEQ    DD
01210 0248 B6 010A           LDA A  OPRNDB
01220 024B BD 0395           JSR    CNVT
01230 024E B6 010A           LDA A  OPRNDB
01240 0251 BD 03CC           JSR    CHAR
01250 0254 A7 03             STA A  3,X
01260 0256 B6 010B    DD     LDA A  OPINDX
01270 0259 16                TAB
01280 025A CE 05D7           LDX    #OPCODE
01290 025D FF 0106           STX    TEMP
01300 0260 48                ASL A              OPx2
01310 0261 24 03             BCC    EE
01320 0263 7C 0106           INC    TEMP        JMP
01330 0266 1B         EE     ABA                OPx3
01340 0267 24 03             BCC    FF          CLC?
01350 0269 7C 0106           INC    TEMP        CLC?
01360 026C BB 0107    FF     ADD A  TEMP+1      OPCODE+3*OP
01370 026F 24 03             BCC    GG
01380 0271 7C 0106           INC    TEMP
01390 0274 B7 0107    GG     STA A  TEMP+1
01400 0277 FE 0106           LDX    TEMP        X = OPCODE + 3*OP
01410 027A A6 00             LDA A  X           get OPCODE LETTER
1420 027C B7 014C            STA A  LINE+21
01430 027F A6 01             LDA A  1,X
01440 0281 B7 014D           STA A  LINE+22
01450 0284 A6 02             LDA A  2,X
01460 0286 B7 014E           STA A  LINE+23
01470 0289 C6 20             LDA B  #'
01480 028B B6 010C           LDA A  OPINFO
01490 028E 85 C0             BIT A  #$C0
01500 0290 27 08             BEQ    REGNO
01510 0292 2A 04             BPL    REGB
01520 0294 C6 41             LDA B  #'A
01530 0296 20 02             BRA    REGNO
01540 0298 C6 42      REGB   LDA B  #'B
01550 029A F7 014F    REGNO  STA B  LINE+24
01560 029D CE 0152           LDX    #LINE+27
01570 02A0 B6 014C           LDA A  LINE+21
01580 02A3 81 2A             CMP A  #'*
01590 02A5 27 4D             BEQ    NOTREL-2    ?
01600 02A7 B6 010C           LDA A  OPINFO
01610 02AA 85 02             BIT A  #2
01620 02AC 27 46             BEQ    NOTREL-2    ?
01630 02AE B6 0108           LDA A  SAVEOP
01640 02B1 81 8D             CMP A  #$8D
01650 02B3 27 06             BEQ    REL
01660 02B5 84 F0             AND A  #$F0
01670 02B7 81 20             CMP A  #$20
01680 02B9 26 3B             BNE    NOTREL
01690 02BB C6 2A      REL    LDA B  #'*
01700 02BD E7 00             STA B  X
01710 02BF 08                INX
01720 02C0 C6 2B             LDA B  #'+
01730 02C2 B6 0109           LDA A  OPRNDA
01740 02C5 4C                INC A
01750 02C6 4C                INC A
01760 02C7 2A 02             BPL    NMREL
01770 02C9 C6 2D             LDA B  #'-
01780 02CB E7 00      NMREL  STA B  X
01790 02CD 08                INX
01800 02CE C6 24             LDA B  #'$
01810 02D0 E7 00             STA B  X
1820 02D2 08                 INX
01830 02D3 BD 0395           JSR    CNVT
01840 02D6 C6 20             LDA B  #'
01850 02D8 E7 00             STA B  X
01860 02DA 08                INX
01870 02DB C6 24             LDA B  #'$
01880 02DD E7 00             STA B  X
01890 02DF 08                INX
01900 02E0 4F                CLR A
01910 02E1 F6 0109           LDA B  OPRNDA
01920 02E4 2A 01             BPL    NNREL
01930 02E6 4A                DEC A
01940 02E7 FB 0103    NNREL  ADD B  ADDR+1
01950 02EA B9 0102           ADC A  ADDR
01960 02ED BD 0395           JSR    CNVT
```

Continued on page 10

complete array with spaces

Continued from page 9

```
01970 02F0 17               TBA
01980 02F1 BD 0395          JSR    CNVT
01990 02F4 20 41            BRA    ILLIGT
02000 02F6 81 80   NOTREL   CMP A  #$80
02010 02F8 27 04            BEQ    IMM
02020 02FA 81 C0            CMP A  #$C0
02030 02FC 26 05            BNE    NOIMM
02040 02FE C6 23   IMM      LDA B  #'#
02050 0300 E7 00            STA B  X
02060 0302 08               INX
02070 0303 B7 0108 NOIMM    STA A  SAVEOP
02080 0306 86 24            LDA A  #'$
02090 0308 A7 00            STA A  X
02100 030A 08               INX
02110 030B B6 0109          LDA A  OPRNDA
02120 030E BD 0395          JSR    CNVT
02130 0311 B6 010C          LDA A  OPINFO
02140 0314 85 01            BIT A  #1
02150 0316 27 06            BEQ    NOOPR
02160 0318 B6 010A          LDA A  OPRNDB
02170 031B BD 0395          JSR    CNVT
02180 031E B6 0108 NOOPR    LDA A  SAVEOP
02190 0321 81 60            CMP A  #$60
02200 0323 27 08            BEQ    INDX
02210 0325 81 A0            CMP A  #$A0
02220 0327 27 04            BEQ    INDX
02230 0329 81 E0            CMP A  #$E0
02240 032B 26 0A            BNE    ILLIGT
02250 032D 86 2C   INDX     LDA A  #',
02260 032F A7 00            STA A  X
02270 0331 08               INX
02280 0332 86 58            LDA A  #'X
02290 0334 A7 00            STA A  X
02300 0336 08               INX
02310 0337 86 0D   ILLIGT   LDA A  #$0D
02320 0339 A7 00            STA A  X
02330 033B 86 0A            LDA A  #$0A
02340 033D A7 01            STA A  1,X
02350 033F 86 04            LDA A  #4
02360 0341 A7 02            STA A  2,X
02370 0343 CE 0137          LDX    #LINE
02380 0346 BD 03C5          JSR    PRINT
02390 0349 86 20            LDA A  #'
02400 034B F6 010C          LDA B  OPINFO
02410 034E C4 10            AND B  #$10
02420 0350 27 02            BEQ    NOLABL
02430 0352 86 2E            LDA A  #'.
02440 0354 B7 0149 NOLABL   STA A  LINE+18
02450 0357 B6 0102          LDA A  ADDR
02460 035A B1 0104          CMP A  DONE
02470 035D 2D 0A            BLT    GOBACK
02480 035F 22 0B            BHI    FINIS
02490 0361 B6 0103          LDA A  ADDR+1
02500 0364 B1 0105          CMP A  DONE+1
02510 0367 22 03            BHI    FINIS
02520 0369 7E 01C1 GOBACK   JMP    REPEAT
02530 036C 86 15   FINIS    LDA A  #21
02540 036E C6 20            LDA B  #'
02550 0370 CE 0137          LDX    #LINE
02560 0373 E7 00   CLR      STA B  X
02570 0375 08               INX
02580 0376 4A               DEC A
02590 0377 26 FA            BNE    CLR
02600 0379 CE 0137          LDX    #LINE
02610 037C 86 45            LDA A  #'E
02620 037E A7 15            STA A  21,X
02630 0380 86 4E            LDA A  #'N
02640 0382 A7 16            STA A  22,X
02650 0384 86 44            LDA A  #'D
02660 0386 A7 17            STA A  23,X
02670 0388 86 0C            LDA A  #$C
02680 038A A7 18            STA A  24,X
2690 038C 86 04             LDA A  #4
02700 038E A7 19            STA A  25,X
02710 0390 8D 33            BSR    PRINT
02720 0392 7E E0E3          JMP    MONIT
02730 0395 36      CNVT     PSH A
02740 0396 8D 06            BSR    CBHLH
02750 0398 32               PUL A
02760 0399 08               INX
02770 039A 8D 06            BSR    CBHRH
02780 039C 08               INX
02790 039D 39               RTS
02800 039E 44      CBHLH    LSR A
02810 039F 44               LSR A
02820 03A0 44               LSR A
02830 03A1 44               LSR A
02840 03A2 84 0F   CBHRH    AND A  #$0F
02850 03A4 8B 30            ADD A  #'0
02860 03A6 81 39            CMP A  #'9
02870 03A8 23 02            BLS    CBHOK
02880 03AA 8B 07            ADD A  #7
02890 03AC A7 00   CBHOK    STA A  X
02900 03AE 39               RTS
02910 03AF B7 801C GO       STA A  PRINTR
02920 03B2 86 36            LDA A  #$36
02930 03B4 B7 801D          STA A  PRINTR+1
02940 03B7 86 3E            LDA A  #$3E
02950 03B9 B7 801D          STA A  PRINTR+1
02960 03BC 7D 801D PLUP     TST    PRINTR+1
02970 03BF 2A FB            BPL    PLUP
02980 03C1 B6 801C          LDA A  PRINTR
02990 03C4 08               INX
03000 03C5 A6 00   PRINT    LDA A  X
03010 03C7 81 04            CMP A  #4
03020 03C9 26 E4            BNE    GO
03030 03CB 39               RTS
03040 03CC 81 20   CHAR     CMP A  #'
03050 03CE 2D 04            BLT    BAD
03060 03D0 81 7F            CMP A  #$7F
03070 03D2 2F 02            BLE    OK
03080 03D4 86 20   BAD      LDA A  #'
03090 03D6 39      OK       RTS
03100 03D7 0001    OPS      FDB    1,$101,1,1,1,1,$201
03110 03E5 0301             FDB    $301,$401,$501,$601,$701
03120 03EF 0801             FDB    $801,$901,$A01,$B01,$C01
03130 03F9 0D01             FDB    $D01,1,1,1,1,$E01,$F01
03140 0407 0001             FDB    1,$1001,$1101,1,1,1,1
03150 0417 1212             FDB    $1212,1,$1302,$1402
03160 041F 1502             FDB    $1502,$1602,$1702,$1802
03170 0427 1902             FDB    $1902,$1A02,$1B02,$1C02
03180 042F 1D02             FDB    $1D02,$1E02,$1F02,$2002
03190 0437 2101             FDB    $2101,$2201,$2381,$2341
03200 043F 2401             FDB    $2401,$2501,$2681,$2641
03210 0447 0001             FDB    1,$2711,1,$2811,1,1,$2901
03220 0455 2A01             FDB    $2A01,$2B81,1,1,$2C81
03230 045F 2D81             FDB    $2D81,1,$2E81,$2F81,$3081
03240 0469 3181             FDB    $3181,$3281,1,$3381,$3481
```

SAMPLE DISASSEMBLY

```
03AF  B7801C        STAA  $801C
03B2  8636     6    LDAA  #$36
03B4  B7801D        STAA  $801D
03B7  863E     >    LDAA  #$3E
03B9  B7801D        STAA  $801D
03BC  7D801D   }    TST   $801D
03BF  2AFB     *    BPL   *-$FD $03BC
03C1  B6801C        LDAA  $801C
03C4  08            INX
03C5  A600          LDAA  $00,X
03C7  8104          CMPA  #$04
03C9  26E4     &    BNE   *-$E6 $03AF
03CB  39       9    RTS
03CC  8120          CMPA  #$20
03CE  2D04     -    BLT   *+$06 $03D4
03D0  817F          CMPA  #$7F
03D2  2F02     /    BLE   *+$04 $03D6
03D4  8620          LDAA  #$20
03D6  39       9    RTS
03D7  00            ***
03D8  01            NOP
03D9  01            NOP
03DA  01            NOP
03DB  00            ***
03DC  01            NOP
03DD  00            ***
03DE  01            NOP
03DF  00            ***
03E0  01            NOP
                    END
```

```
03250 0473 0001             FDB    1,$3581,$2B41,1,1,$2C41
03260 047F 2D41             FDB    $2D41,1,$2E41,$2F41,$3041
03270 0489 3141             FDB    $3141,$3241,1,$3341
03280 0491 3341             FDB    $3341,1,$3541,$2B02,1,1
03290 049D 2C02             FDB    $2C02,$2D02,1,$2E02,$2F02
03300 04A7 3002             FDB    $3002,$3102,$3202,1,$3302
03310 04B1 3402             FDB    $3402,$3612,$3502,$2B03
03320 04B9 0001             FDB    1,1,$2C03,$2D03,1,$2E03
03330 04C5 2F03             FDB    $2F03,$3003,$3103,$3203,1
03340 04CF 3303             FDB    $3303,$3403,$3613,$3503
03350 04D7 3782             FDB    $3782,$3882,$3982,1,$3A82
03360 04E1 3B82             FDB    $3B82,$3C82,1,$3D82,$3E82
03370 04EB 3F82             FDB    $3F82,$4082,$4103,$4202
03380 04F3 4303             FDB    $4303,1,$3782,$3882
```


Continued on page 11

Continued from page 11

```
04FB 3982          FDB    $3982,1,$3A82,$3B82,$3C82
0505 4482          FDB    $4482,$3D82,$3E82,$3F82
050D 4082          FDB    $4082,$4102,1,$4302,$4502
0517 3782          FDB    $3782,$3882,$3982,1,$3A82
0521 3B82          FDB    $3B82,$3C82,$4482,$3D82
0529 3E82          FDB    $3E82,$3F82,$4082,$4102
0531 4602          FDB    $4602,$4302,$4502,$3783
0539 3883          FDB    $3883,$3983,1,$3A83
0541 3B83          FDB    $3B83,$3C83,$4483,$3D83
0549 3E83          FDB    $3E83,$3F83,$4083,$4103
0551 4603          FDB    $4603,$4303,$4503,$3742
0559 3842          FDB    $3842,$3942,1,$3A42,$3B42
0563 3C42          FDB    $3C42,1,$3D42,$3E42
056B 3F42          FDB    $3F42,$4042,1,1,$4703
0575 0001          FDB    1,$3742,$3842,$3942,1
057F 3A42          FDB    $3A42,$3B42,$3C42,$4442
0587 3D42          FDB    $3D42,$3E42,$3F42,$4042
058F 0001          FDB    1,1,$4702,$4802,$3742,$3842
059B 3942          FDB    $3942,1,$3A42,$3B42
05A3 3C42          FDB    $3C42,$4442,$3D42,$3E42
05AB 3F42          FDB    $3F42,$4042,1,1,$4702
05B5 4802          FDB    $4802,$3743,$3843,$3943
05BD 0001          FDB    1,$3A43,$3B43,$3C43
05C5 4443          FDB    $4443,$3D43,$3E43,$3F43
05CD 4043          FDB    $4043,1,1,$4703,$4803
05D7 2A     OPCODE FCC    '***NOPTAPTPAINXDEXCLVSEVCLCSECCLISEI      missing 10/1/78

05FB 53            FCC    'SBACBATABTBADAAABABRABHIBLSBCCBCSBNE BEQ BVC

0625 42            FCC    'BVSBPLBMIBGEBLTBGTBLETSXINSPULDESTXS PSH RTS

064F 52            FCC    'RTIWAISWINEGCOMLSRRORASRASLROLDECINC' TST CLR

0679 4A            FCC    'JMPSUBCMPSBCANDBITLDAEORADCORAADDCPX BSR LDS

06A3 53            FCC    'STASTSJSRLDXSTX

06BE 4D            FCC    'MODIFIED BY T. J. WEAVER'

                   END

TOTAL ERRORS 00000
```


LIST OF REFERENCES

1. Hughes, P. *Introducting the Disassembler.* KILOBAUD, 7 (60) 1077.

Captain Thomas J. Weaver, USAF
825 N. Sherry
Norman, Oklahoma 73069

The author is a doctorate student majoring in computer science at the University of Oklahoma.

HEX/ASCII DUMP PROGRAM

Those of us who have worked with Assembler Language on large IBM systems have been exposed to the IBM Assembler Dump. (At least, those of us who are not perfect.) One of the nice things about the IBM dump is the character dump that is provided in addition to the hex portion of the dump. This can be very useful, since character strings can give quite valuable hints as to the location of portions of our programs or clues as to exactly when or why they bombed. Because of their importance the author decided to expend a small amount of effort writing a HEX/ASCII DUMP PROGRAM.

The program outputs the address followed by the hex data, grouped to make it easier to find a specific address. This is then followed by a dump of those same locations in ASCII. Where the content of the byte under consideration are not printable, the character / is substituted.

The program is not commented (a bad habit), so it will be explained as to its workings for those who might want to modify it. It is assembled at 0000 to make it easier for those who want to hand assemble their own version to move it to whatever location suits their systems. Mine is assembled at $DO8O and functions as a transient command under the Smoke Signal Broadcasting Disk Operating System.

Upon entry, the program prompts for the beginning location of the dump. This is then rounded down to the next lower number ending in 0. This is used as the actual starting address of the dump, in order to make it more readable. The ending address is then obtained and rounded up in a similar fashion. The line number is then output using the OUT4HS routine from MIKBUG®. Two additional spaces are thrown in for readability. A pair of nested loops is then used to format the hex portion of the dump in groups of four. After 16 bytes are output, the index is reset to the beginning of the line, and the ASCII portion of the dump begins. The bytes are picked up one at a time and compared with the upper and lower limits of the ASCII printable character set. If these limits are exceeded, control transfers to the routine NO which places the substitute character / in the A register in place of the unprintable character. This character may be easily changed to any other character at the discretion of the individual programmer. When the program reaches the ending address, the program transfers control back to MIKBUG.

The best way to use this program is to add it to whatever operating system you are using. It may be reassembled or you might use the dump to enter it directly, changing the locations that refer to relocated storage. I hope you find it useful.

```
                            NAM DUMP
                            OPT SYM,NOG,LIS
E0CA               OUT2HS   EQU $E0CA
E0C8               OUT4HS   EQU $E0C8
E1D1               OUTEE    EQU $E1D1
E07E               PDATA1   EQU $E07E
E047               BADDR    EQU $E047
0000                        ORG 0
0000 CE 00 A7      START    LDX #SADD
0003 BD E0 7E               JSR PDATA1
0006 BD E0 47               JSR BADDR
0009 FF 00 CF               STX TEMP
000C B6 00 D0               LDAA TEMP+1
000F 84 F0                  ANDA #$F0
0011 B7 00 D0               STAA TEMP+1
0014 CE 00 BC               LDX #EADD
0017 BD E0 7E               JSR PDATA1
001A BD E0 47               JSR BADDR
001D FF 00 D1               STX XHIGH
0020 B6 00 D2               LDAA XHIGH+1
0023 8B 10                  ADDA #$10
0025 84 F0                  ANDA #$F0
0027 B7 00 D2               STAA XHIGH+1
002A 24 07                  BCC SKIP
002C B6 00 D1               LDAA XHIGH
002F 4C                     INCA
0030 B7 00 D1               STAA XHIGH
0033 BD 00 9C      SKIP     JSR CRLF
0036 BD E1 D1               JSR OUTEE
0039 CE 00 CF      LOOP     LDX #TEMP
003C BD E0 C8               JSR OUT4HS
003F 86 20                  LDAA #$20
0041 BD E1 D1               JSR OUTEE
0044 BD E1 D1               JSR OUTEE
0047 FE 00 CF               LDX TEMP
004A C6 04         HEXLP1   LDAB #4
004C BD E0 CA      HEXLP2   JSR OUT2HS
004F 5A                     DECB
```

Continued on page 12

- Continued from page 11

```
0050 26 FA              BNE HEXLP2
0052 86 20              LDAA #$20
0054 BD E1 D1           JSR OUTEE
0057 7A 00 D3           DEC COUNT
005A 26 EE              BNE HEXLP1
005C 86 20              LDAA #$20
005E BD E1 D1           JCR OUTEE
0061 BD E1 D1           JSR OUTEE
0064 FF 00 CF           STX TEMP
0067 B6 00 D0           LDAA TEMP+1
006A 80 10              SUBA #$10
006C B7 00 D0           STAA TEMP+1
006F 24 03              BCC SSKIP
0071 7A 00 CF           DEC TEMP
0074 C6 10       SSKIP  LDAB #16
0076 FE 00 CF           LDX TEMP
0079 A6 00       ASCLP  LDAA X
007B 81 1F              CMPA #$1F
007D 23 19              BLS NO
007F 81 7E              CMPA #$7C
0081 22 15              BHI NO
0083 BD E1 D1    BACK   JSR OUTEE
0086 08                 INX
0087 5A                 DECB
0088 26 EF              BNE ASCLP
008A FF 00 CF           STX TEMP
008D BD 00 9C           JSR CRLF
0090 BC 00 D1           CPX XHIGH
0093 26 A4              BNE LOOP
0095 7E E0 E3           JMP $E0E3
0098 86 5C       NO     LDAA #'\
009A 20 E7              BRA BACK
009C 86 0D       CRLF   LDAA #$0D
009E BD E1 D1           JSR OUTEE
00A1 86 0A              LDAA #$0A
00A3 BD E1 D1           JSR OUTEE
00A6 39                 RTS
00A7 0D 0A       SADD   FDB $0D0A
00A9 53                 FCC /STARTING ADDRESS: /
00BB 04                 FCB 4
00BC 0D 0A       EADD   FDB $0D0A
00BE 45                 FCC /ENDING ADDRESS: /
00CE 04                 FCB 4
00CF             TEMP   RMB 2
00D1             XHIGH  RMB 2
00D3             COUNT  RMB 1
                        END
      NO ERROR(S) DETECTED


   SYMBOL TABLE:
ASCLP   0079      BACK    0083      DADDR   E047      COUNT   00D3
CRLF    009C      EADD    00BC      HEXLP1  004A      HEXLP2  004C
LOOP    0039      NO      0098      OUT2HS  E0CA      OUT4HS  E0C8
OUTEE   E1D1      PDATA1  E07E      SADD    00A7      SKIP    0033
SSKIP   0074      START   0000      TEMP    00CF      XHIGH   00D1


 DUMP

STARTING ADDRESS: 0000
ENDING ADDRESS: 00D3
0000  CC 00 A7 BD  E0 7E DD E0  47 FF 00 CF  B6 00 D0 84   \\\\\\~\\G\\\\\\\\
0010  F0 B7 00 D0  CE 00 BC BD  E0 7E BD E0  47 FF 00 D1   \\\\\\\\\\\~\\G\\\\
0020  B6 00 D2 8B  10 84 F0 B7  00 D2 24 07  B6 00 D1 4C   \\\\\\\\\\\\$\\\\\L
0030  B7 00 D1 BD  00 9C BD E1  D1 CE 00 CF  BD E0 C8 86   \\\\\\\\\\\\\\\\\\
0040  20 BD E1 D1  BD E1 D1 FE  00 CF C6 04  BD E0 CA 5A    \\\\\\\\\\\\\\\\Z
0050  26 FA 86 20  BD E1 D1 7A  00 D3 26 EE  86 20 BD E1   &\\ \\\z\\&\\ \\
0060  D1 BD E1 D1  FF 00 CF B6  00 D0 80 10  B7 00 D0 24   \\\\\\\\\\\\\\\\\$
0070  03 7A 00 CF  C6 10 FE 00  CF A6 00 81  1F 23 19 81   \z\\\\\\\\\\\\#\\
0080  7E 22 15 BD  E1 D1 08 5A  26 EF FF 00  CF BD 00 9C   ~"\\\\\\Z&\\\\\\\\
0090  BC 00 D1 26  A4 7E E0 E3  86 5C 20 E7  86 0D BD E1   \\\&\~\\\\ \\\\\\
00A0  D1 86 0A BD  E1 D1 37 0D  0A 53 54 41  52 54 49 4E   \\\\\\7\\STARTIN
00B0  47 20 41 44  44 52 45 53  53 3A 20 04  0D 0A 45 4E   G ADDRESS: \\\EN
00C0  44 49 4E 47  20 41 44 44  52 45 53 53  3A 20 04 20   DING ADDRESS: \
00D0  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20
```

BFD-68-2 Smoke Signal Broadcasting®, are Registered Trademarks and Copyrights of the Smoke Signal Broadcasting Corporation, Hollywood, California.
MIKBUG® is a registered Trademark of Motorola, Incorporated.

Captain Paul Dobbs, USAF
4348 Brom Bones Alley
Fort Worth, Texas 76114

DISC TEXT PROCESSING

If text processing is your 'thing' and you have Smoke Signal Broadcasting's BDF-68® floppy disc system and Technical Systems Consultants® SL68-29, v.2.3 text processor program, then here is an offer you can't (*hardly*) refuse. After all its free. Its 'disc text processing'. Adding this program to the SL68-29 allows the text processor to call text from a disc file with the only restriction on the length of the processed text being the storage space available on the disc. Total ram memory requirement, regardless of text size, is 9584 (decimal) bytes. All features of the TSC TEXT PROCESSOR are retained.

After loading the TSC text processor and 'disc' programs into ram the program should be saved as a transient command program as follows:

SAVE, DTEXT, 200,20CA, 2003, $ (CR)

Calling text files into the processor is accomplished through either of two methods. The first is to include the text 'filename' when calling the processor separating the two files with a comma.

DTEXT, TEXT FILENAME (CR)

The second method is particularly useful when you forget to use the first and, with single disc systems, when the text and processor are on different discs. To use it call the disc processor from disc followed by a carrage return.

DTEXT (CR)

The processor will run as usual up to where actual text processing begins. At this point the program requests a file name which the user supplies followed by a carrage return.

FILE NAME: TEXT FILENAME (CR)

Initial program entry has been changed to 2003 (hex) to allow resetting of flags required by the disc patches. After processing, or in the event of a system error, control is returned to the DOS monitor program through ZWARMS after printing an error

Continued on page 13

Continued from page 12

message and type number if any are detected. Extensive use has been made of subroutines in the DOS. Smoke Signal has provided jump points for these routines as well as an excellent description of the functioning of each in the BFD-68 operating manual. A little time spent on this portion of the manual will provide a good deal of help in understanding the inner workings of the DOS as well as the basis for writing your own disc system programs.

One final note, the disc program uses a 1-K buffer memory for storing text from disc prior to processing. The length of this storage was chosen somewhat arbitrarily and may be shortened or lengthened as desired by changing the buffer end address specified in the program at hex addresses 2014 and 2015 (BUFST+$400). The shorter the buffer memory the more frequently will it require refilling from disc.

```
                        DISC TEXT PROCESSOR
                        ---- ---- ---------

                PATCHES TO ALLOW TSC TEXT PROCESSOR
                VERSION 2.3 TO OPERATE FORM SMOKE
                SIGNAL BROADCASTING DISC SYSTEM.
1972            MACTBL    EQU     $1972
0212            INTRO     EQU     $212
00FA            MACEND    EQU     $FA
0097            FSTRAM    EQU     $97
0099            LSTRAM    EQU     $99
009B            NXTRAM    EQU     $9B
009D            JNKCNT    EQU     $9D
14D6            INCHR2    EQU     $14D6
7283            ZWARMS    EQU     $7283
7291            ZFLSPC    EQU     $7291
7294            ZGCHAR    EQU     $7294
7297            ZGNCHR    EQU     $7297
729A            ZANCHK    EQU     $729A
729D            ZDIE      EQU     $729D
72A0            ZGETHN    EQU     $72A0
72A3            ZADDX     EQU     $72A3
72A6            ZOUTST    EQU     $72A6
72A9            ZTYPDE    EQU     $72A9
72B5            ZLINEI    EQU     $72B5
7780            ODFM      EQU     $7780
7783            CDFM      EQU     $7783
7786            DFM       EQU     $7786

2000                      ORG     $2000

2000 00         BTEMP     FCB     0
2001 00 00      XTEMP     FDB     0

                ENTRY POINT FOR DISC VERSION

2003 7F 00 38   DISCST    CLR     RPFLG      CLEAR REPEAT FLAG
2006 7E 02 12             JMP     INTRO      ENTER MAIN PROC PROGRAM

2009 CE 19 72   DWORD     LDX     #MACTBL
200C DF FA                STX     MACEND
200E CE 21 71   PROCST    LDX     #BUFST     START OF BUFFER MEMORY
2011 DF 97                STX     FSTRAM     END OF BUFFER
2013 CE 25 71             LDX     #BUFST+$400
2016 DF 99                STX     LSTRAM
2018 86 00                LDA A   #0         SET JUNK COUNT - FOR CORES
201A 97 9D                STA A   JNKCNT     EDITOR SET TO 5
201C 7D 00 38             TST     RPFLG      .RP INSTRUCTION?
201F 27 05                BEQ     DWORD2
2021 B7 00 38             STA A   RPFLG      CLEAR FLAG
2024 20 18                BRA     RETN
2026 BD 72 94   DWORD2    JSR     ZGCHAR     LOOK FOR A COMMA
2029 81 2C                CMP A   #',        IF NONE ASK FOR
202B 27 09                BEQ     SKIP       FILE NAME
202D CE 20 B9             LDX     #MSG
2030 BD 72 A6             JSR     ZOUTST     PRINT MSG
2033 BD 72 B5             JSR     ZLINEI     GET FILE NAME
2036 CE 20 CB   SKIP      LDX     #FCB
2039 BD 72 91             JSR     ZFLSPC     LOAD FCB WITH FILE NAME
203C 25 2F                BCS     ERR1       CHECK FOR DFM ERRORS
203E BD 20 45   RETN      JSR     OPEN       OPEN FILE
2041 BD 20 76             JSR     FILBF      STORE IN BUFFER
2044 39                   RTS
                OPEN FILE FOR READING

2045 86 04      OPEN      LDA A   #4         OPEN FILE MODE
2047 CE 20 CB             LDX     #FCB       LOAD FILE CONTROL BLOC IN X
204A A7 00                STA A   0,X        STORE IN XFC OF THE FCB
204C BD 77 86             JSR     DFM        REQUEST DFM OPEN FILE
204F 26 1C                BNE     ERR1       LOOK FOR ERRORS AND PRINT #
```


Continued on page 14

Continued from page 13

```
2051 86 05                 LDA A  #5
2053 A7 00                 STA A  0,X         SET FCB TO READ MODE
2055 39                    RTS
                READ FROM OPENED FILE

2056 FF 20 01   READ       STX    XTEMP       SAVE INDEX
2059 CE 20 CB              LDX    #FCB
205C BD 77 86              JSR    DFM         REQUEST DFM ENTER READ
205F 27 08                 BEQ    NOERR       CHECK FOR DFM ERRORS
2061 A6 01                 LDA A  1,X         CHECK ERROR TYPE
2063 81 06                 CMP A  #6          FOR EOF (#6)
2065 26 06                 BNE    ERR1        IF NOT EOF PRINT ERROR #
2067 86 1A                 LDA A  #$1A        LOAD PROC EOF IN REG A
2069 FE 20 01   NOERR      LDX    XTEMP       RESTORE X
206C 39                    RTS
                ERROR REPORTING ROUTINE

206D BD 72 A9   ERR1       JSR    ZTYPDE      PRINT ERROR TYPE #
2070 BD 77 83   ERR2       JSR    CDFM        CLOSE ALL OPEN FILES
2073 7E 72 83              JMP    ZWARMS      RETURN TO MONITOR
                FILL BUFFER FROM DISC

2076 7F 00 37   FILBF      CLR    FIN         CLEAR FIN FLAG
2079 DE 97                 LDX    FSTRAM      LOAD START OF BUFFER
207B DF 9B                 STX    NXTRAM      NEXT RAM ADDR
207D 9C 99      FILL1      CPX    LSTRAM      BUFFER FULL?
207F 27 17                 BEQ    FILL4
2081 8D D3                 BSR    READ        GET MORE DATA FROM DISC
2083 81 1A                 CMP A  #$1A        LOOK FOR PROC EOF
2085 27 05                 BEQ    FILL2
2087 A7 00                 STA A  0,X         FILL BUFFER
2089 08                    INX
208A 20 F1                 BRA    FILL1
208C B7 00 37   FILL2      STA A  FIN
208F 09         FILL3      DEX
2090 A6 00                 LDA A  0,X         BACK UP LOOK FOR LAST CR
2092 81 0D                 CMP A  #$D
2094 26 F9                 BNE    FILL3
2096 DF 99                 STX    LSTRAM      SET BUFFER END
2098 DE 97      FILL4      LDX    FSTRAM
209A 39                    RTS
                REFILL BUFFER FROM DISC

209B F7 20 00   BUFRFL     STA B  BTEMP       SAVE B
209E 7D 00 37              TST    FIN         EOF?
20A1 26 0B                 BNE    END1
20A3 BD 20 76              JSR    FILBF
20A6 F6 20 00              LDA B  BTEMP
20A9 86 00                 LDA A  #0          CLEAR A OF GARBAGE
20AB 7E 14 D6   OUT1       JMP    INCHR2      RETURN TO PROC
20AE 86 1A      END1       LDA A  #$1A        SET PROC EOF CONT.Z
20B0 20 F9                 BRA    OUT1
                .RP INSTRUCTION RELOAD FROM TOP

20B2 BD 77 83   RESTRT     JSR    CDFM        CLOSE FILES
20B5 7C 00 38              INC    RPFLG       SET REPEAT FLAG
20B8 39                    RTS

20B9 0D         MSG        FCB    $D,$A,4,4,4
20BA 0A 04
20BC 04 04
20BE 46                    FCC    /FILE NAME : /
20BF 49 4C
20C1 45 20
20C3 4E 41
20C5 4D 45
20C7 20 3A
20C9 20
20CA 00                    FCB    0
                FILE CONTROL BLOC

20CB            FCB        RMB    166

                BUFFER STORAGE

2171            BUFST      RMB    $3FF
2570                       RMB    $1          END OF BUFFER FYI
                FLAG STORAGE

0037                       ORG    $37
0037            FIN        RMB    1           DISC FILE END
0038            RPFLG      RMB    1           REPEAT FLAG

                PATCHES IN TSC TEXT PROCESSOR PROGRAM V.2.3

0209                       ORG    $209        EXIT TO MONITOR
0209 7E 20 70              JMP    ERR2        CLOSE FILES RETURN TO MON

14E1                       ORG    $14E1       FILL BUFFER & SET EOF
14E1 7E 20 9B              JMP    BUFRFL      FILL BUFFER
```

Continued on page 15

Continued from page 14

```
14E5                  ORG    $14E5     RESTART FROM START
14E5 BD 20 82  RWND   JSR    RESTRT    .RP INSTRUCTION
14E8 39               RTS

0338                  ORG    $338      DISC SETUP
0338 BD 20 09         JSR    DWORD
033B 01               NOP    FILL
033C 01               NOP

                      END

SYMBOL TABLE:
BTEMP   2000    BUFRFL  209B    BUFST   2171    CDFM    7783
DFM     7786    DISCST  2003    DWORD   2009    DWORD2  2026
END1    20AE    ERR1    206D    ERR2    2070    FCB     20CB
FILBF   2076    FILL1   207D    FILL2   208C    FILL3   208F
FILL4   2098    FIN     0037    FSTRAM  0097    INCHR2  14D6
INTRO   0212    JNKCNT  009D    LSTRAM  0099    MACEND  00FA
MACTBL  1972    MSG     20B9    NOERR   2069    NXTRAM  009B
ODFM    7780    OPEN    2045    OUT1    20AB    PROCST  200E
READ    2056    RESTRT  20B2    RETN    203E    RPFLG   0038
RWND    14E5    SKIP    2036    XTEMP   2001    ZADDX   72A3
ZANCHK  729A    ZDIE    729D    ZFLSPC  7291    ZGCHAR  7294
ZGETHN  72A0    ZGNCHR  7297    ZLINE1  72B5    ZOUTST  72A6
ZTYPDE  72A9    ZWARMS  7283
```

R. Lynn Smith
7713 Glenister Drive
Springfield, Virginia. 22152

BDF-68® Disc Operating System are Copyrighted and Registered Trademarks of the Smoke Signal Broadcasting Corporation, Hollywood, California.
TSC® Text Processing System and FLEX® Operating System are Copyrighted and Registered Trademarks of the Technical Systems Consultants Incorporated, West Lafayette, Indiana.

BALIBAGO DOUBLE STANDARD

Or a Hardwired Approach to Adding 1200 Baud to the AC-30® Cassette Interface

Ok, we've got our Mikbug® and out TVT-II running at 1200 baud. Why not go all the way as we used to say and also bring the AC-30® up to full speed? Actually its not a very difficult feat, and for results — how does loading 8K basic in about 75 seconds sound?

Unfortunately, as with so many good ideas this one was not original with us. Vic Bolstad and I were inspired to work on a hardware based 1200 baud cassette system by Helmers. Helmers (1) describes a software approach to operating an unmodified (AC-30® at 1200 bps which is interesting but not entirely consistent with the Mikbug® based mode of operation we were used to. Arriving almost the same day was Huffman's description (2) of increasing the speed of Mikbug® by increasing the clock frequency feeding the MP-C control interface board (also refer Hilbun (3). What Vic and I really wanted was to operate our systems at both 1200 and 300 bps, switch selectable from the front panel. What we came up with was a surprisingly simple arrangement we call the "*Balibago Double Standard.*" In keeping with the practice established by BYTE Magazine we named it after the city where it was developed, Balibago, Pampanga, Republic of the Philippines, a small recreational and retirement community in Central Luzon.

One of the features of the *Kansas City Standard* which I personally feel has been greatly over exaggerated in importance is the self-clocking feature. This is used to derive an approximate (depending on the tape speed) 4800 Hz signal to be used as a clock frequency during read operations. The idea at first glance is good but, my experience has been that, at least with the AC-30®, it causes more trouble than it cures. My disconnecting this line and connecting the 6800 MP-C board transmitter clock line directly to the

Continued on page 16

Continued from page 15

MP-C receiver clock input has resulted in consistently better copy. Cassette speed variation does not appear to be the problem the designers of the *Kansas City Standard* thought it would be. Besides, removing the self-clocking feature makes using the AC-30® at 1200 Baud much easier. The same procedure should be followed with the TVT-II UART's receiver clock input. A feature that should be added also is to feed the TVT-II UART clock inputs from the MP-C data clock. Doing this allows you to switch the speed of everything from 300 to 1200 (or other rates below 1200) at the same time and with only one switch.

ADDITIONAL CIRCUITY

At 300 baud a mark represents eight cycles at 2400 Hz, a space four cycles at 1200 Hz. This provides a lot of room for slop, and the lack of phase coherency between data and tone frequencies other than causing a tolerable increase signal distortion may be ignored. This is most definitely not the case at 1200 baud where we have only one cycle at 1200 Hz in which to recognize a space. What this boils down to is the need to assure that the start (zero crossing) of a mark or space tone burst occurs reasonably close to mark-space transistions coming from the computer.

Phase synchronization to within acceptable limits is acheived by utilizing the 76.8 KHz output available as one of the unused clock frequencies from the Motorola MC14411 bit rate generator on the SWTPC® 6800 MPU board. In my system I connected this output (pin 2 of the bit rate generator) to the 150 baud line (any unused line will do) with a short jumper in order to get it onto the bus and into the control interface board where it then through the interconnecting cable becomes the AC-30® clock input. Of course, disconnected the 150 baud line is disconnected from the bit rate generator (pin 8) first. In the AC-30® this signal is first divided by either two or four depending on the output data and then by 16 down to the 2400 or 1200 Hz tones seen by the recorder. Synchronization with the mark-space transistions to within *plus or minus 1/16th* of a cycle is achieved by resetting the 7493 to zero each time the control interface data output switches from high (mark) to low (space). Since the same tone frequencies are used at both 1200 and 300 baud the divider circuit remains in for operation at either speed.

DATA READ.

Although the demodulator used in the AC-30® leaves some room for improvement when operated at 1200 baud, no changes other than disconnecting the read clock as discussed above, should be necessary in order to read data recorded at either 1200 or 300 baud. Remember it is necessary for the clock frequency to match the read data rate feeding into the UART receiver input.

If there is further interest, perhaps is a future article we can discuss some possible improvements to the demodulator in the AC-30®.

CONSTRUCTION

Since only two additional IC's are involved construction is an easy matter. The AC-30® supplies the necessary + 5 VDC. Simply mounting the two IC's on a small universal PC board on stand-offs above the main AC-30® board in a convenient spot works out well and by lifting the end of R2 that ties to pin 1 of IC5A the counter input and output may be connected to the AC-30® board. A jumper wire to Pin 1 of IC2 supplies data information for resetting the counter.

R. Lynn Smith
7713 Glenister Drive
Springfield, Virginia 22152

List of References

1. Helmers, C. *A Software Controlled 1200 BPS Audio Tape Interface* 2 (41) 1977, Byte.
2. Huffmann, J. *Speed Up Your 6800.* 4 (49) 1977, Kilobaud.
3. Hilbun, R.L. *Megabaud 1.2 Maybe* Journal of the 6800 International Computer Center Director.